Introducción al Aprendizaje Automático con YOLO
Introduction to Machine Learning with YOLO

Raquel Miranda Pérez
Instituto Tecnológico de Costa Rica
Miranda.p.raquel@gmail.com
Jaffette Solano Arias
Instituto Tecnológico de Costa Rica
jaffette@estudiantec.cr
Abel Méndez Porras
Instituto Tecnológico de Costa Rica
amendez@tec.ac.cr fpicado@uned.ac.cr

Recibido 18/mar/2019
Aprobado 20/may/2019

Resumen-
El reconocimiento de objetos en imágenes ha sido una tarea tradicionalmente compleja para los sistemas computacionales y la dificultad se incrementa si se desea realizar en tiempo real. Con los avances del aprendizaje automático se han desarrollado algoritmos que permiten el reconocimiento de objetos en tiempo real con una latencia bastante baja. El objetivo de este artículo es proporcionar una guía práctica a los usuarios que desean iniciar en el reconocimiento de objetos utilizando el sistema YOLO (You Only Look Once). Se hace una descripción de YOLO. Se presenta una herramienta de etiquetado de objetos para la creación de patrones de entrenamiento. Se explica el proceso de entrenamiento de YOLO utilizando Darknet. Se muestra la forma de utilizar YOLO para reconocer objetos. Además, se describen una serie de proyectos encontrados en la literatura que implementaron una solución basada en YOLO.

Palabras Claves: *Aprendizaje automático, YOLO, BBox-Tool, patrones de entrenamiento, reconocimiento de objetos.*

Abstract Context:
The recognition of objects in images has been a traditionally complex task for computer systems and the difficulty is increased if you want to do it in real time. With the advances in machine learning algorithms has been possible to develop the recognition of objects in real time with a very low latency. The objective of this article is to contribute to users who wants to start recognizing objects using YOLO (You Only Look Once) through a practical guide. A description of YOLO is made. A tool for labeling objects for the creation of training patterns is presented. The training process of YOLO using Darknet is explained. The way to use YOLO to recognize objects is shown. In addition, a series of projects found in the literature that implemented a solution based on YOLO are described.

Keywords: *Machine learning, YOLO, Box-Tool, training patterns, object recognition.*

I. INTRODUCCIÓN

El análisis de imágenes es una forma de emprender en el área del machine learning. Existen variedad de herramientas que se pueden utilizar para detectar objetos en imágenes y videos, unas más laboriosas que otras, pero todas con la capacidad de adaptarse según las necesidades y preferencias.

El objetivo de este artículo es proporcionar una guía práctica a los usuarios que desean iniciar en el reconocimiento de objetos utilizando el sistema YOLO. En este caso se describe el sistema de detección de objetos YOLOv3 mediante Darknet, como una herramienta que posee un modelo ya pre-entrenado que permite detectar variedad de objetos en imágenes y videos en tiempo real. Este modelo se puede modificar para crear un modelo personalizado, es ahí donde inician los aportes de este documento, procurando ser una guía práctica de cómo empezar a generar productos que se integren en el campo del Machine Learning, como el reconocimiento de objetos en imágenes.

Este documento está estructurado de la siguiente manera: la sección II presenta proyectos encontrados en la literatura que emplean YOLO como parte de la solución, la sección III presenta una guía práctica para iniciar con el reconocimiento de objetos utilizando el sistema YOLO, por último, la sección IV se concluye sobre las posibilidades de utilizar YOLO para el reconocimiento de objetos.

II. PROYECTOS EN LA LITERATURA DESARROLLADOS UTILIZANDO YOLO

Mugahed et al. (A.Al-antaria, A.Al-masnia, Choib, Hana, & Kima, 2018) implementaron YOLO para la detección y clasificación del cáncer de mama. Hamed and Avaznia (Hamed Naghavi, Avaznia, & Talebi, 2018) hicieron una versión conjunta de Fast R-CNN con YOLO para detectar objetos en imágenes capturadas por vehículos de conducción automática, concluyeron que su enfoque es más lento que YOLO, pero más rápido que Fast R-CNN. El diseño de regresión rápida en YOLO puede ser combinado con Deep CNN para la detección y la segmentación semántica simultánea en las carreteras (Teichmann, Weber, Zoellner, Cipolla, & Urtasun, 2018). Hsu, Ambikapathi, Chung and Su (Hsu, Ambikapathi, & Su, 2017) diseñaron un detector de matrículas de vehículos utilizando YOLO y YOLOv2. Hsieh, Lin, and Hsu (Meng-Ru , Yen-Liang , & Winston H. , 2017) utilizaron YOLO y Fast R-CNN para contar los autos presentes en un estacionamiento utilizando un drone. YOLO ha sido probado con colecciones de imágenes rastreadas en la web (Jeong, Park, & Ha, 2018), y ha sido sujeto de cambios en su código fuente. YOLO contiene muchas clases para la clasificación de objetos y tiene fallas en el posicionamiento y en el enfoque de entrenamiento, así como deficiencias en la detección de objetos muy pequeños, por lo tanto, se han buscado soluciones que optimicen la velocidad, precisión y consumo de energía en los sistemas de reconocimiento. Una opción es utilizar Tiny YOLO, una versión de solo 28MB que es mucho más rápida y liviana (Kang, Kang, Kang, Yoo, & Ha, 2018), investigadores como Li, Chen and ChaoYang (Nguyen, y otros, 2018) hicieron uso de Tiny YOLO en los procesadores de tolerancia a fallas y determinaron que rendimiento es equivalente al mejor rendimiento actual del procesador, otros investigadores también diseñaron frameworks utilizando YOLO como base (Yan, Chen, Chen, Kendrick, & Wu, 2018).

Du (Du, 2018) menciona en su artículo que YOLOV2 provee el mejor equilibrio y

precisión que la primera versión de YOLO. Otros modelos para la detección de objetos como Tensorflow y Caffe 1.0, se han probado para comparar su rendimiento con YOLO (Fulop & Tamas, 2018).

III. APRENDIZAJE AUTOMÁTICO UTILIZANDO YOLOV3

En esta sección se presenta una guía práctica para iniciar con el reconocimiento de objetos utilizando el sistema YOLO. Se describe el sistema de reconocimiento de objetos YOLO, la configuración para Darknet, la forma de crear los patrones de entrenamiento, entrenar un modelo, probar el modelo y detectar objetos en imágenes en tiempo real con el modelo entrenado en YOLOv3.

A. Descripción de YOLO

YOLO es una sola red convolucional que predice simultáneamente múltiples cuadros delimitadores y probabilidades de clase para esos cuadros. Usa características de toda la imagen para predecir cada cuadro delimitador. También predice todos los cuadros delimitadores en todas las clases para una imagen simultáneamente. YOLO divide la imagen de entrada en una cuadrícula S × S. Si el centro de un objeto se ubica en una celda de la cuadrícula, esa celda es responsable de detectar ese objeto. YOLO se entrena con imágenes completas y optimiza directamente el rendimiento de detección. Además, puede procesar video en tiempo real con menos de 25 milisegundos de latencia.

La Figura 1 muestra la arquitectura de YOLO. La red de detección tiene 24 capas convolucionales seguidas de 2 capas completamente conectadas. La alternancia de capas convolucionales 1×1 reduce el espacio de características de las capas anteriores. Se pre-entrenan las capas convolucionales en la tarea de clasificación de ImageNet a la mitad de la resolución (imagen de entrada 224×224) y luego se duplica la resolución para la detección.

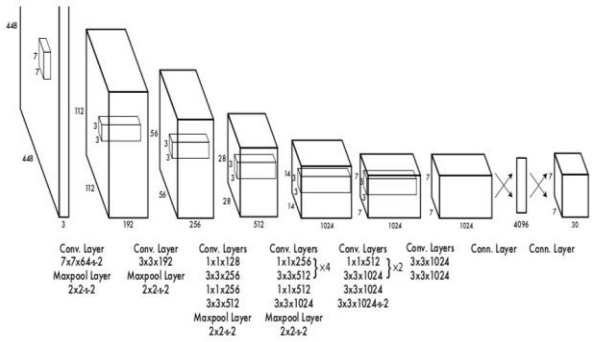

Figura 1. Arquitectura del sistema YOLO. Figura tomada de [1].

En la página oficial del YOLO https://pjreddie.com/darknet/yolo/ se puede encontrar suficiente información para descargar y utilizar este sistema de detección de objetos. Una versión de Darknet utilizando YOLO puede ser descargada de la siguiente ubicación:

git clone https://github.com/pjreddie/darknet
cd darknet make

Descargar los pesos pre-entrenados de la siguiente ubicación:

wget https://pjreddie.com/media/files/yolov3.weights

B. Configuración para Darknet

Previa a la instalación de Darknet se debe tomar en cuenta que puede ser ejecutado en sistemas operativos como Windows, Mac OS y Linux. Otro de los puntos a considerar es que puede ser ejecutado tanto para aquellas computadoras que cuenten con tarjeta gráfica dedicada como aquellas que no lo hacen, no obstante, el proceso de entrenamiento y reconocimiento será mucho más rápido en aquellas que sí cuenten con una tarjeta gráfica. De contar con una tarjeta gráfica podremos instalar CUDA que se puede descargar desde https://developer.nvidia.com/cuda-

downloads. Es de suma importancia la configuración del archivo "make" pues, en él se colocará un uno o cero según la configuración de la computadora. Para efectos de este documento los comandos e instrucciones se llevaron a cabo en Ubuntu (distribución de Linux) y con una tarjeta gráfica (Nvidia 1080 Ti)

C. Crear patrones de entrenamiento

La herramienta BBox-Tool permite crear patrones para entrenar modelos creados utilizando la biblioteca de YOLOv3. BBox-Tool se puede utilizar para etiquetar cuadros delimitadores de objetos en imágenes. Esta herramienta esta implementada en Python Tkinter y se puede descargar desde: https://github.com/chinkan/BBox-Label-Tool-For-Yolo.

Figura 2. Etiquetando un objeto para crear un patrón de entrenamiento.

En la Figura 2 se puede apreciar un objeto etiquetado utilizando la herramienta BBox-Tool. El rectángulo de color rojo representa el objeto etiquetado, en este caso, una bola de tenis. Para crear un nuevo cuadro de límite o etiquetado de un objeto, se hace clic izquierdo para seleccionar el primer vértice. Se mueve el ratón para dibujar un rectángulo y se hace clic nuevamente para seleccionar el segundo vértice.

D. Entrenar un modelo de machine learning en YOLOv3

Una vez las imágenes han sido previamente etiquetadas con la herramienta BBox-Tool, esta herramienta producirá una serie de archivos por imagen en una carpeta llamada labels. Estos archivos son de tipo (.txt) y contienen las coordenadas de la imagen en el que se encuentra el objeto que se quiere reconocer. No obstante, el formato generado por el programa para etiquetar no es capaz de ser leído por YOLO. Se debe convertir en un formato reconocido por YOLO mediante un algoritmo escrito en Python. Se toman todas estas coordenadas y se transcribe, además se reubica los nuevos archivos de tipo (.txt) junto a cada imagen en una carpeta. Una vez se tienen las imágenes con sus respectivos archivos (.txt) estos deberán de ser colocados en la carpeta data, misma que se localiza en la raíz de Darknet. Previo a que las imágenes hayan sido movidas, se debe crear 2 archivos, uno que contendrá el nombre del objeto a reconocer llamado object.names y el otro con las ubicaciones de donde se guardaran pesos y las ubicaciones de las imágenes para crear el set de entrenamiento llamado object.data.

Una vez que se tiene todos estos pasos completos se debe descargar un archivo de la página oficial de Darknet YOLOv3 con los pesos para del modelo de machine learning. El archivo contiene los pesos predeterminados antes de empezar a entrenar, pero se debe tomar en cuenta que hay más de un archivo y que se debe de seleccionar el que corresponda a la versión de YOLO que se está utilizando.

Una vez descargado el archivo se procede a entrenar, para lo que se ejecuta el siguiente comando en una terminal:

./darknet detector train data/obj.data yolov3-obj.cfg "archivo preentrenado descargado".

La Figura 3 muestra el proceso de entrenamiento de Darknet. Este proceso es lento y preferiblemente utilizar un GPU (Unidad de Procesamiento Grafico) para

mejorar el desempeño. A continuación, se describe los componentes del comando anterior:

- darknet: este archivo ejecutable de Darknet.
- detect: este comando indica que se va a detectar objetos en una imagen.
- train: este comando indica que se va a entrenar el modelo de machine learning.
- data/obj.data: ubicación de los patrones de entrenamiento.
- yolov3-obj.cfg: configuración de YOLO.

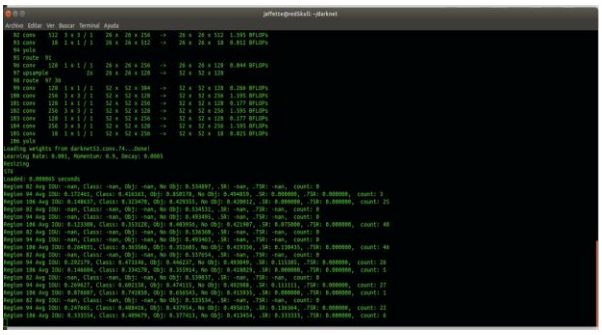

Figura 3. Proceso de entrenamiento de un modelo de machine learning.

E. Reconocer objetos

Una vez que un modelo creado en YOLOv3 ha sido entrenado utilizando Darknet, se puede utilizar para reconocer objetos. Un ejemplo del comando que se debe ejecutar para realizar el reconocimiento de objetos es el siguiente:

./darknet detect cfg/yolov3.cfg yolov3.weights data/bola.jpeg

A continuación, se describe los componentes del comando anterior:

- darknet: este archivo es el que permite ejecutar el modelo entrenado.
- detect: este comando indica que se va a detectar objetos en una imagen.
- yolov3.cfg: este archivo contiene la información de la configuración del modelo que se ha entrenado.
- yolov3.weights: este archivo contiene los valores de los pesos ajustados después del entrenamiento del modelo.
- bola.jpeg: este archivo es la imagen en la que se desea reconocer objetos.

Figura 4. Objeto reconocido utilizando el modelo entrenado.

En la Figura 4 se puede apreciar la forma en que el modelo entrenado reconoce el objeto tipo bola a partir de una imagen. El objeto reconocido es etiquetado con un cuadro y el nombre de la categoría a la que pertenece, en este ejemplo, “sports ball”.

Es importante mencionar que si existen más objetos de la misma categoría también serán reconocidos y etiquetados. Además, si existen objetos de otras categorías que hayan sido parte del entrenamiento también serán reconocidos y etiquetados.

Darknet también permite realizar reconocimiento de objetos en tiempo real. Si desea reconocer objetos del video obtenido de la cámara se puede utilizar el siguiente comando:

./darknet detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights

Si se desea reconocer un objeto de un video previamente almacenado, se puede utilizar el siguiente comando:

./darknet detector demo cfg/coco.data cfg/yolo.cfg yolo.weights videofile.mp4

A continuación, se describe los componentes del comando anterior:

- darknet: este archivo es el que permite ejecutar Darknet.

- detector: este comando indica que se va a detectar objetos desde una cámara o video.
- cfg/coco.data: indica que se va a utilizar el conjunto de datos de COCO.
- yolov3.cfg: este archivo contiene la información de la configuración del modelo que se ha entrenado.
- yolov3.weights: este archivo contiene los valores de los pesos ajustados después del entrenamiento del modelo.
- Video-file.mp4: en el caso de que se desea reconocer objetos desde un video, se agrega el nombre del archivo de video.

IV. CONCLUSIONES

En este documento ofrece una guía práctica para iniciar en el reconocimiento de objetos con el sistema YOLOv3. Se hizo una descripción de YOLO y se presentó una herramienta de etiquetado de objetos para la creación de patrones de entrenamiento. También se explicó el proceso de entrenamiento de YOLO utilizando Darknet y la forma de utilizar YOLO para reconocer objetos en imágenes, videos o desde una cámara en tiempo real. Además, se describieron una serie de proyectos encontrados en la literatura que implementaron una solución basada en YOLO.

YOLO procesa imágenes en tiempo real a 45 marcos por segundo y aprende representaciones muy generales de objetos. Es un detector de objetos rápido y preciso, lo que lo hace ideal para aplicaciones de visión artificial.

Incluso, ofrece buenas prestaciones procesando imágenes de videos en tiempo real.

Los avances en aprendizaje automático permiten crear sistemas de reconocimiento de objetos en tiempo real muy rápidos y precisos. Sin embargo, para un buen funcionamiento de estos sistemas se requiere del uso de GPU's potentes. Lo anterior dificulta la extender este tipo de tecnologías a mayor escala.

IV. REFERENCIAS

A.Al-antaria, M., A.Al-masnia, M., Choib, M.-T., Hana, S.-M., & Kima, T.-S. (2018). A fully integrated computer-aided diagnosis system for digital X-ray mammograms via deep learning detection, segmentation, and classification. *International Journal of Medical Informatics*, 44-54. doi:https://doi.org/10.1016/j.ijmedinf.2018.06.003

Du, J. (2018). Understanding of Object Detection Based on CNN Family and YOLO. *Journal of Physics: Conference Series. 1004*, 1-8. doi:10.1088/1742-6596/1004/1/012029

Fulop, A.-O., & Tamas, L. (2018). Lessons learned from lightweight CNN based object recognition for mobile robots. *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 1-5. Obtenido de https://ieeexplore.ieee.org/document/8402778

Hamed Naghavi, S., Avaznia, C., & Talebi, H. (2018). Integrated real-time object detection for self-driving vehicles. *Computer and Knowledge Engineering (ICCKE) 2018 8th International Conference on*, 274-279. Obtenido de https://ieeexplore.ieee.org/document/8342340/citations?tabFilter=papers#citations

Hsu, G.-S., Ambikapathi, A., & Su, C.-P. (2017). Robust license plate detection in the wild. *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference*, 1-6. Obtenido de

https://ieeexplore.ieee.org/document/8078493/authors#authors

Jeong, H.-J., Park, K.-S., & Ha, Y.-G. (2018). Image Preprocessing for Efficient Training of YOLO Deep Learning Networks. *Big Data and Smart Computing (BigComp)*, 635–637. Obtenido de https://ieeexplore.ieee.org/document/8367193

Kang, D., Kang, D., Kang, J., Yoo, S., & Ha, S. (2018). Joint optimization of speed, accuracy, and energy for embedded image recognition systems. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 715–720. Obtenido de https://ieeexplore.ieee.org/document/8342102

Meng-Ru , H., Yen-Liang , L., & Winston H. , H. (2017). Drone-based Object Counting by Spatially Regularized Regional Proposal Network. *IEEE International Conference on Computer Vision (ICCV)*. Obtenido de https://arxiv.org/abs/1707.05972

Nguyen, P., Arsalan, M., Koo, J., Naqvi, R., Truong, N., & Park, K. (2018). LightDenseYOLO: A Fast and Accurate Marker Tracker for Autonomous UAV Landing by Visible Light Camera Sensor on Drone. *Sensors*, 2-30. doi:https://www.mdpi.com/1424-8220/18/6/1703

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., & Urtasun, R. (2018). MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. *IEEE Intelligent Vehicles Symposium*, 1013–1020. Obtenido de https://arxiv.org/abs/1612.07695

Yan, C., Chen, W., Chen, P., Kendrick, A., & Wu, X. (2018). A new two-stage object detection network without RoI-Pooling. *Chinese Control And Decision Conference (CCDC)*, 1680–1685. Obtenido de https://ieeexplore.ieee.org/document/8407398